

What is static block?

A class level nameless block that contains only static keyword in its declaration is called static block.

Syntax for creating static block

```
class Example {  
    static {  
        -----  
        -----  
        -----  
        -----  
    }  
}
```

} except return and
throw statement all
statements are allowed

Why static block?

It is used for initializing static variables and some logic only once at the time of class loading.

Different ways to execute logic at the time of class loading

Actually there are two ways to execute logic at the time of class loading.

1. Using static variable.

```
class Example{  
    static int a = m1();  
  
    static int m1(){  
        System.out.println("SV : a");  
        return 10;  
    }  
    public static void main(String[] args) {  
        System.out.println("main");  
        m1();  
    }  
}
```

Two drawbacks in this approach

1. m1() method logic can also be executed after class loading, because we can call it from main() method, that leads to execution of m1() after class loading.
 2. For calling method we must create static variable, which leads to extra memory and extra time will be consumed.

2. Using static block

Write the logic in static block, because it does not have name, we cannot call it, the logic is executed only once.

Who will execute static block when and where?

Static blocks are executed automatically by JVM at the time of class loading in the order they defined from top to bottom before main method or before the calling member.

Then what is order of execution of SB and Main method?

Static block is always executed before main method.

What is the output from the below program?

```
class Example {  
    static {  
        System.out.println("SB");  
    }  
    public static void main(String[] args) {  
        System.out.println("main");  
    }  
}
```

```
class Example {  
    public static void main(String[] args) {  
        System.out.println("main");  
    }  
    static {  
        System.out.println("SB");  
    }  
}
```

Below program explains initializing static variable in static block & reading it from main method

```
class Example{  
    static int a;  
    static {  
        System.out.println("SB");  
        a = 50;  
    }  
    public static void main(String[] args) {  
        System.out.println("main");  
        System.out.println(" a: "+a);  
    }  
}
```

```
class Test27_SB_TC8{  
    static int a;  
    static {  
        System.out.println("SB");  
        int a = 50;  
    }  
    public static void main(String[] args) {  
        System.out.println("main");  
        System.out.println(" a: "+a);  
    }  
}
```

```
class Example{  
    static int a;  
    static {  
        System.out.println("SB");  
        int a = 50;  
        a = 60;  
    }  
    public static void main(String[] args) {  
        System.out.println("main");  
        System.out.println(" a: "+a);  
    }  
}
```

```
class Example{  
    static int a;  
    static {  
        System.out.println("SB");  
        int a = 50;  
        a = 60;  
        Example.a = 70;  
    }  
    public static void main(String[] args) {  
        System.out.println("main");  
        System.out.println(" a: "+a);  
    }  
}
```

Below program explains initializing static variable in static block with runtime value

```
import java.util.Scanner;
class Example{
    static int a;

    static {
        System.out.println("SB");
        Scanner scn = new Scanner(System.in);

        System.out.print("Enter a: ");
        a = scn.nextInt();
        System.out.println("value is stored in SV a\n");

    }
    public static void main(String[] args) {
        System.out.println("main");
        System.out.println(" a: "+a);
    }
}
```

How many static blocks can be defined in a class?

In a class we can define multiple static blocks.

What is the order of execution of all static blocks?

All static blocks are executed in the order they defined from top to bottom.

Check below program, what is the output?

```
class Example {
    static {
        System.out.println("SB1");
    }
    static {
        System.out.println("SB2");
    }
    public static void main(String[] args) {
        System.out.println("main");
    }
}
```

```
class Example {
    static {
        System.out.println("SB2");
    }
    public static void main(String[] args) {
        System.out.println("main");
    }
    static {
        System.out.println("SB1");
    }
}
```

What is the need of multiple static blocks?

For developing modularity for initializing each static variable separately

Below program explains need of multiple static blocks

```
class Example {  
    static int a;  
    static {  
        System.out.println("SB1");  
        a = 10;  
    }  
    static int b;  
    static {  
        System.out.println("SB2");  
        b = 20;  
    }  
    public static void main(String[] args) {  
        System.out.println("main");  
        System.out.println("a: "+a);  
        System.out.println("b: "+b);  
    }  
}
```

Can we nest static blocks means Can we write a static block inside another static block?

No, static keyword is not allowed inside blocks or methods, it leads to compile time error.

```
class Example {  
    static {  
        System.out.println("SB1");  
  
        static {  
            System.out.println("SB2");  
        }  
    }  
    public static void main(String[] args) {  
        System.out.println("main");  
    }  
}
```

Q) Can we execute main method at the time of class loading?

Yes, it is possible. Call it from static block it is also executed at the time of class loading.

What is the output from the below program?

```
class Example {  
    static {  
        System.out.println("SB ");  
        main(new String[0]);  
    }  
    public static void main(String[] args){  
        System.out.println("main");  
    }  
}
```

O/P

SB
main
main